Software Safety

Nancy G. Leveson

MIT
Aeronautics and Astronautics
leveson@mit.edu

http://sunnyday.mit.edu

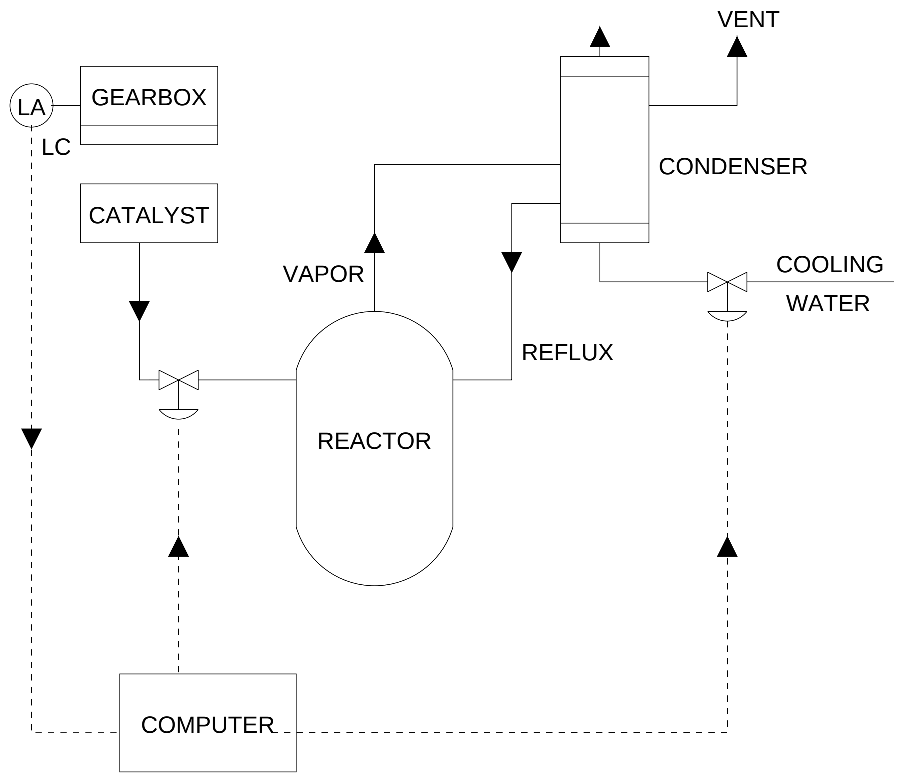
Presentation at the FAA Software Certification Meeting, Danvers, MA June 7 2001

Conclusions

- Safety $\neq$ Reliability
- Most software-related accidents stem from requirements flaws
- Safety must be built-in

  Requires additions and changes to standard processes

Accident with No Component Failures



Types of Accidents

- Component Failure Accidents

  Single or multiple component failures

  Usually assume random failure

- System Accidents

  Arise in interactions among components

  No components may have "failed"

  Caused by interactive complexity and tight coupling

  Exacerbated by the introduction of computers.

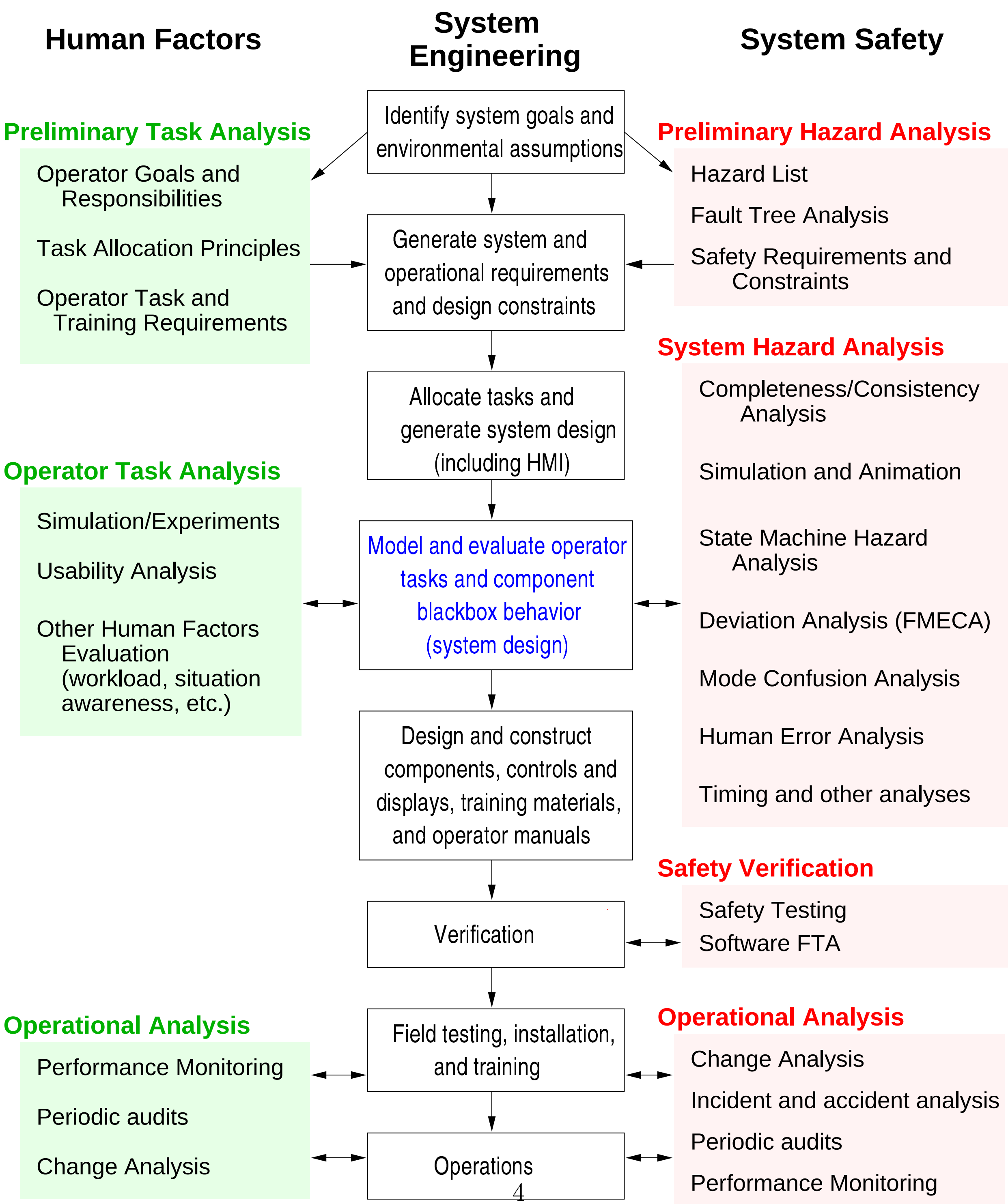
Software-Related Accidents

- Are usually caused by flawed requirements
  - Incomplete or wrong assumptions about operation of controlled system or required operation of computer.
  - Unhandled controlled-system states and environmental conditions.
- Merely trying to get the software ‘‘correct’’ or to make it reliable will not make it safer.

Software-Related Accidents (con’t.)

- Software may be highly reliable and ‘‘correct’’ and still be unsafe.
  - Correctly implements requirements but specified behavior unsafe from a system perspective.
  - Requirements do not specify some particular behavior required for system safety (incomplete)
  - Software has unintended (and unsafe) behavior beyond what is specified in requirements.
- In highly-automated systems, accidents have changed their nature.

A Human-Centered, Safety-Driven Design Process



Management Changes

- Software safety as part of system safety plan
- Assigning responsibility and authority

    software safety engineer?
- Software hazard tracking system
- Communication channels
- Anomaly reporting system

Process Steps

1. Preliminary Task Analysis

    Operator goals and responsibilities

    Task allocation principles

    Operator task and training requirements

    Preliminary Hazard Analysis

    Hazard list (hazard not equal to failure)

    System hazard analysis (tracing hazards to potential causes)

    Safety requirements and constraints

Specifying Safety Constraints

- Most software requirements only specify nominal behavior

  Need to specify off-nominal behavior

  Need to specify what software must NOT do

- What must not do is not inverse of what must do

- Derived from system hazard analysis (not failure analysis)

Process Steps (2)

2. Generate system requirements and design constraints using information in PTA and PHA

3. Design at system level to eliminate or control hazards

4. Trace hazards and hazard controls to software

Process Steps (3)

5. Software requirements review and analysis

   Completeness

   Simulation and animation

   Software hazard analysis

   Robustness (environment) analysis

   Mode confusion and other human error analyses

   Human factors analyses (usability, workload, etc.)

Process Steps (4)

6. Implementation with safety in mind

   Defensive programming

   Assertions and run-time checking

   Separation of critical functions

   Elimination of unnecessary functions

   Exception-handling etc.

7. Off-nominal and safety testing

Process Steps (5)

8. Operational Analysis and Auditing

    Change analysis

    Incident and accident analysis

    Performance monitoring

    Periodic audits

SpecTRM

Specification Tools and Requirements Methodology

A set of integrated tools to assist in
building complex safety-critical systems.

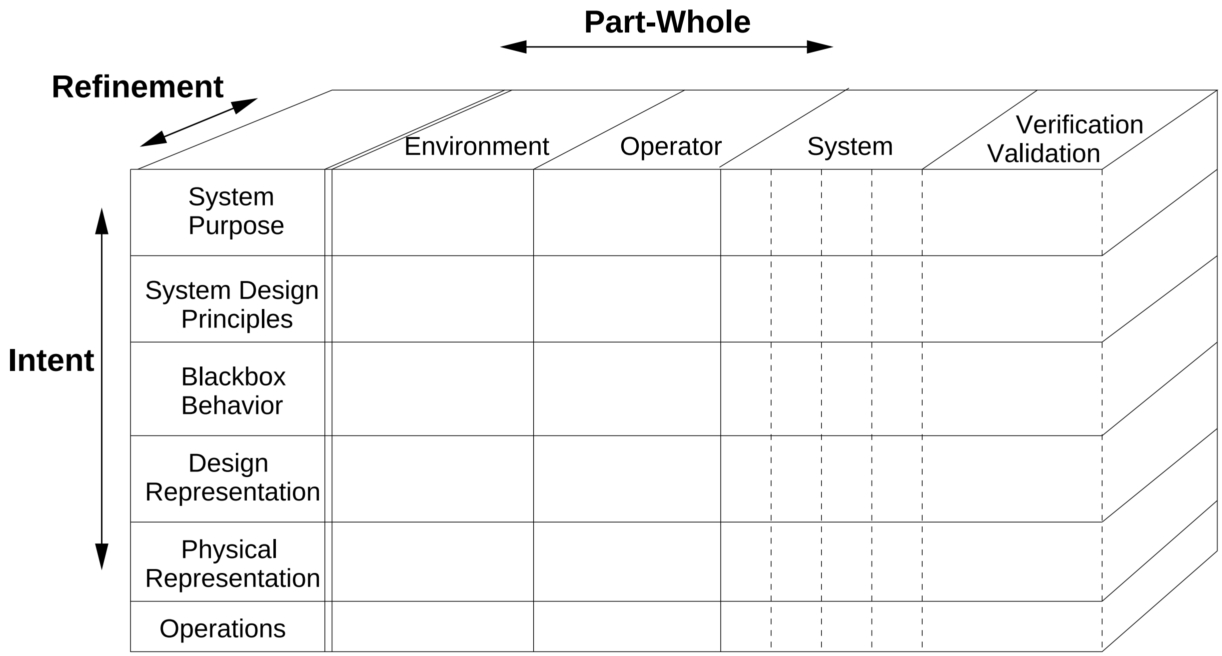
SpecTRM Goals

- Support systems engineering
  - Requirements development
  - Seamless transition from system to software
  - Evolution and maintenance
- Provide human-centered specification and analysis tools
  - Built on knowledge about human-problem solving
  - Uses visualization and new types of abstraction to enhance intellectual manageability.
- Support task-centered automation (vs. technology-centered)
- Provide cost effective safety assurance

Intent Specifications

- Bridge between disciplines
- Support for human problem solving
- Traceability
- Support for safety analyses
- Integration of formal and informal specifications
- Assistance in software evolution
- Hierarchical abstraction based on ''why'' (design rationale) as well as what and how.

Intent Specifications



- Each level supports a different type of reasoning about system.
- Mappings between levels provide relational info necessary to reason across hierarchical levels.

Decomposition (horizontal); **Refinement**; **Intent** (vertical)

	Environment	Operator	System and components	V&V
Level 0	Project management plans, status information, safety plan, etc.			
Level 1 System Purpose	Assumptions Constraints	Responsibilities Requirements I/F requirements	System goals, high-level requirements, design constraints, limitations	Hazard Analysis
Level 2 System Principles	External interfaces	Task analyses Task allocation Controls, displays	Logic principles, control laws, functional decomposition and allocation	Validation plan and results
Level 3 Blackbox Models	Environment models	Operator Task models HCI models	Blackbox functional models Interface specifications	Analysis plans and results
Level 4 Design Rep.		HCI design	Software and hardware design specs	Test plans and results
Level 5 Physical Rep.		GUI design, physical controls design	Software code, hardware assembly instructions	Test plans and results
Level 6 Operations	Audit procedures	Operator manuals Maintenance Training materials	Error reports, change requests, etc.	Performance monitoring and audits

Level 1: System Purpose

- Introduction
- Historical Perspective
- Environment Description
- Environment Assumptions
  - Altitude information is available from intruders with a minimum precision of 100 feet.
  - All aircraft have legal identification numbers.
- Environment Constraints
  - The behavior or interaction of non-TCAS equipment with TCAS must not degrade the performance of the TCAS equipment.
- System Functional Goals
  - Provide affordable and compatible collision avoidance system options for a broad spectrum of National Airspace System users.
- High-Level Requirements

[1.2] TCAS shall provide collision avoidance protection for any two aircraft closing horizontally at any rate up to 1200 knots and vertically up to 10,000 feet per minute.

Assumption: Commercial aircraft can operate up to 600 knots and 5000 fpm during vertical climb or controlled descent (and therfore the planes can close horizontally up to 1200 knots and vertically up to 10,000 fpm.

- Design and Safety Constraints

[SC5] The system must not disrupt the pilot and TCC operations during critical phases of flight nor disrupt aircraft operation.

[SC5.1] The pilot of a TCAS-equipped aircraft must have the option to switch to the Traffic-Advisory-Only mode where TAs are displayed but display of resolution advisories is prohibited.

Assumption: This feature will be used during final approach to parallel runways when two aircraft are projected to come close to each other and TCAS would call for an evasive maneuver.

Example Level 1 Safety Constraints for TCAS

SC-7 TCAS must not create near misses (result in a hazardous level of vertical separation) that would not have occurred had the aircraft not carried TCAS.

SC-7.1 Crossing maneuvers must be avoided if possible.

↓ 2.36, 2.38, 2.48, 2.49.2

SC-7.2 The reversal of a displayed advisory must be extremely rare.

↓ 2.51, 2.56.3, 2.65.3, 2.66

SC-7.3 TCAS must not reverse an advisory if the pilot will have insufficient time to respond to the RA before the closest point of approach (four seconds or less) or if own and intruder aircraft are separated by less than 200 feet vertically when 10 seconds or less remain to closest point of approach.

↓ 2.52

Level 1: System Purpose (2)

- System Limitations

  L.5 TCAS provides no protection against aircraft with nonoperational or non-Mode C transponders.

- Operator Requirements

  OP.4 After the threat is resolved the pilot shall return promptly and smoothly to his/her previously assigned flight path.

- Human-Interface Requirements

- Hazard and other System Analyses

TCAS does not display a resolution advisory.

- TCAS unit is not providing RAs.
    - <Self-monitor shuts down TCAS unit>
    - Sensitivity level set such that no RAs are displayed.
    - . . .
- No RA inputs are provided to the display.
    - No RA is generated by the logic
        - Inputs do not satisfy RA criteria
            - Surveillance puts threat outside corrective RA position.
                - Surveillance does not pass adequate track to the logic
                    - <Threat is non-Mode C aircraft> → L.5
                    - <Surveillance failure> → 1.23.1
                - <Surveillance error causes incorrect range/range rate to be calculated>
            - Altitude reports put threat outside corrective RA position
                - Altitude errors put threat on ground
                    - <Uneven terrain> ↓ 2.19
                    - <Intruder altitude error>
                    - <Own Mode C altitude error> → 1.23.1
                    - <Own radar altimeter error> → 1.23.1
                - Altitude errors put threat in non-threat position.
                - . . .
            - <Intruder maneuver causes logic to delay RA beyond CPA> → SC4.2 ↓ 2.35
        - . . .
    - <Process/display connectors fail>
- <Display is preempted by other functions> → SC4.8 ↓ 2.22
- <Display hardware fails> → 1.23.1

TCAS displays a resolution advisory that the pilot does not follow.

- Pilot does not execute RA at all.
  - Crew does not perceive RA alarm.
    - <Inadequate alarm design> → 1.4 to 1.14 ↓ 2.74, 2.76
    - <Crew is preoccupied>
  - <Crew does not believe RA is correct.> → OP.1
  - ...
- Pilot executes the RA but inadequately
  - <Pilot stops before RA is removed> → OP.10
  - <Pilot continues beyond point RA is removed> → OP.4
  - <Pilot delays execution beyond time allowed> → OP.10

Example Level-2 System Design for TCAS

SENSE REVERSALS: Reversal-Provides-More-Separation ↓3.31

2.51 In most encounter situations, the resolution advisory sense will be maintained for the duration of an encounter with a threat aircraft.

↑ SC-7.2

However, under certain circumstances, it may be necessary for that sense to be reversed. For example, a conflict between two TCAS-equipped aircraft will, with very high probability, result in selection of complementary advisory senses because of the coordination protocol between the two aircraft. However, if coordination communications between the two aircraft are disrupted at a critical time of sense selection, both aircraft may choose their advisories independently.

↑ FTA-1300

This could possibly result in selection of incompatible senses.

↑ FTA-395

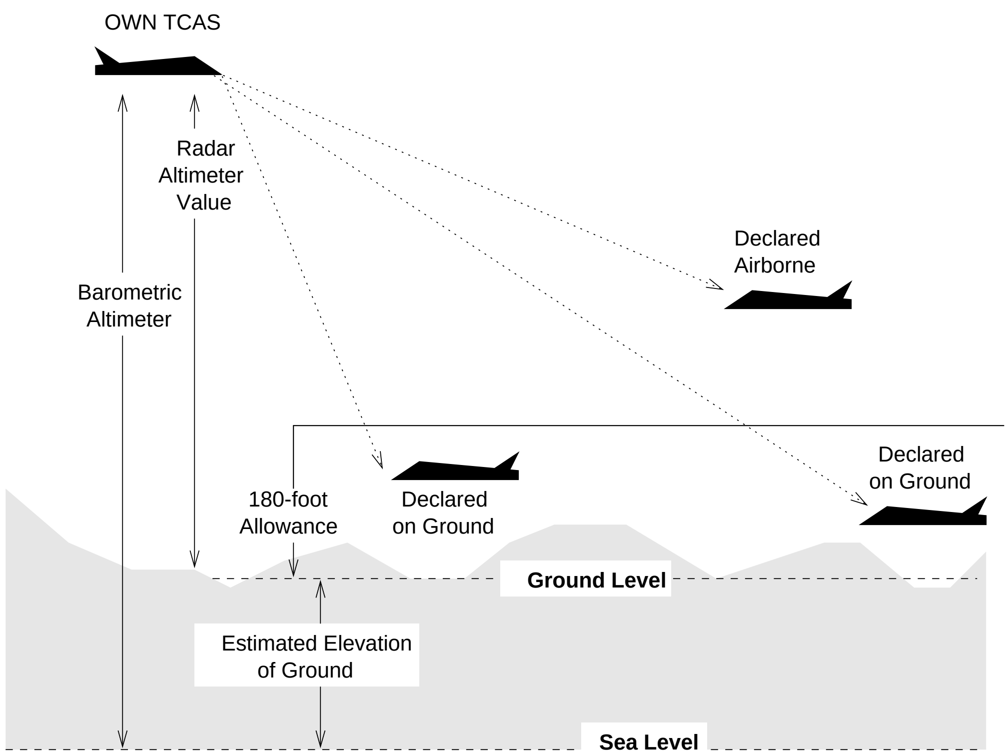
2.51.1 [Information about how incompatibilities are handled]

2.19 When below 1700 feet AGL, the CAS logic uses the difference between its own aircraft pressure altitude and radar altitude to determine the approximate elevation of the ground above sea level (see Figure 2.5).

It then subtracts the latter value from the pressure altitude value received from the target to determine the approximate altitude of the target above the ground (barometric altitude - radar altitude + 180 feet). If this altitude is less than 180 feet, TCAS considers the target to be on the ground (↑ 1.SC4.9).

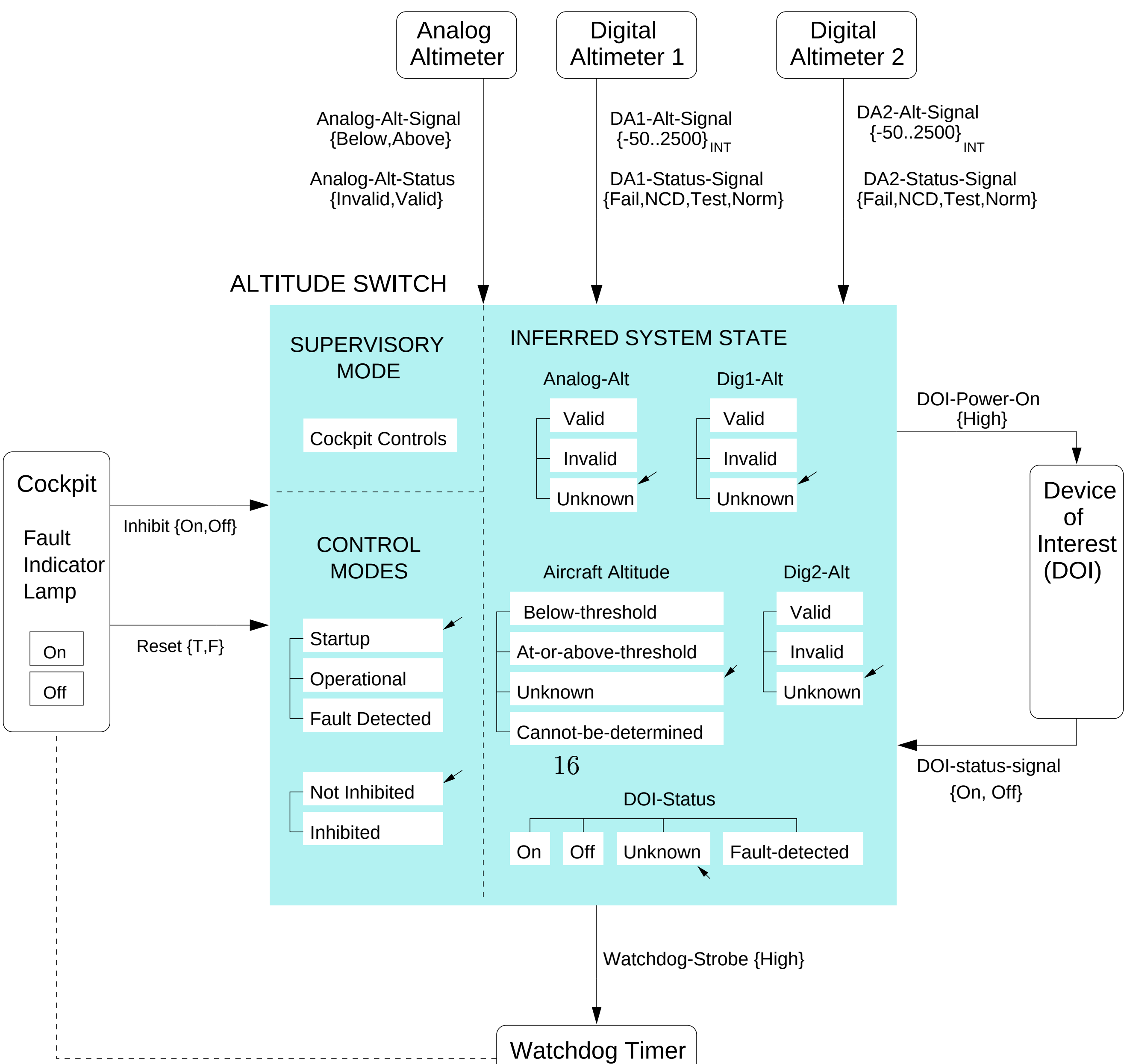
Traffic and resolution advisories are inhibited for any intruder whose tracked altitude is below this estimate. Hysteresis is provided to reduce vacillations in the display of traffic advisories that might result from hilly terrain (↑ FTA-320).

All RAs are inhibited when own TCAS is within 500 feet of the ground.



SpecTRM-RL

- Executable, black-box specifications
  - System components specified only in terms of outputs and the inputs that trigger them.
  - Specify external behavior only—no internal design.
  - System behavior is combined behavior of components.
- Underlying formal model - RSM
  (Requirements State Machine)



Output Command

DOI-Power-On

Destination: DOI

Acceptable Values: {high}

Initiation Delay: 0 milliseconds

Completion Deadline: 50 milliseconds

Exception-Handling: (What to do if cannot issue command within deadline time)

Feedback Information:

Variables: DOI-status-signal

Values: high (on)

Relationship: Should be on if ASW sent signal to turn on

Min. time (latency): 2 seconds

Max. time: 4 seconds

Exception Handling: DOI-Status changed to Fault-Detected

Reversed By: Turned off by some other component or components. Do not know which ones.

Comments: I am assuming that if we do not know if the DOI is on, it is better to turn it on again, i.e., that the reason for the restriction is simply hysteresis and not possible damage to the device.

This product in the family will turn on the DOI only when the aircraft descends below the threshold altitude. Only this page needs to change for a product in the family that is triggered by rising above the threshold.

References: ↑2.4.3 ↓4.7

CONTENTS

= discrete signal on line PWR set to high

TRIGGERING CONDITION

Control Mode	Operational	T
	Not Inhibited	T
State Values	DOI-Status = On	F
	Altitude = Below-threshhold	T
	Prev(Altitude) = At-or-above-threshold	T

Feasibility and Scalability

Models have been or are being built for:

TCAS II

NASA industrial robot

FMS for Draper autonomous helicopter

FMS for MD-11

FMS for experimental NASA aircraft

HETE satellite attitude control

Advanced concepts in ATC

CTAS PFAST (final approach spacing tool)

MTCD (Eurocontrol Conflict Detection Tool)

STARS (Raytheon): only small parts so far

Safety Analysis

- Completeness and consistency analysis
- Simulation and animation
- Operator task analysis
- State machine hazard analysis (backward search)
  Including hybrid modeling and analysis
- Software Deviation Analysis
- Human Error Analysis (Mode Confusion)
- Timing Analysis

Requirements Completeness Analysis

- Most software-related accidents involve software requirements deficiencies.
- Accidents often result from unhandled and unspecified cases.
- We have defined a set of criteria to determine whether a requirements specification is complete.
- Derived from accidents and basic engineering principles.
- Validated (at JPL) and used on industrial projects.

Completeness:	Requirements are sufficient to distinguish the desired behavior of the software from that of any other undesired program that might be designed.

Requirements Completeness Criteria

- Startup, shutdown, mode transitions
- Inputs and outputs
- Robustness
- Value and timing
- Load and capacity
- Environment capacity constraints
- Failure states and transitions
- Human-computer interface
- Data age
- Latency
- Feedback
- Reversibility
- Preemption
- Path Robustness

State Machine Hazard Analysis

- Backward search from hazardous state in SpecTRM-RL model.
- Generates information needed to eliminate hazardous state or to detect and handle it.
- We are now extending models and analysis to include hybrid (discrete plus continuous) system models.

Software Deviation Analysis

- A new type of hazard analysis (Jon Reese's dissertation, 1996) based on the model that accidents are caused by deviations in system variables (e.g., flow, pressure).

  Related to HAZOP and FMECA

- A type of forward robustness analysis: How will software operate in an imperfect environment.
- Determines whether a hazardous software behavior can result from a class of input deviations

  e.g., measured aircraft velocity too low (measured or assumed velocity is less than actual)

Software Deviation Analysis (2)

- Procedure is completely automated. Analyst provides:
  - Input deviations to check
  - SpecTRM-RL (or RSML) specification
  - Safety-critical outputs

- Output is a list of scenarios
  - Set of deviations in software inputs plus paths through specification sufficient to lead to a deviation in a safety-critical output.
  - SDA procedure can optionally add further deviations as it executes that would, together with original deviation, lead to unsafe output.

    Thus allows for analysis of multiple, independent deviations or failures.

Human Error Analysis

Two complementary approaches:

1. Focus on automation:

   Evaluate contribution to human error.

2. Focus on humans:

   Evaluate effect of proposed system changes on human performance.

Operator Task Models

- To ensure safe and efficient operations, must look at the interaction between the human controllers and the computer.
- Break down high-level task into subtasks and model
- Uses same underlying formal modeling language but more appropriate visual representation.
- Operator task models can be executed and analyzed along with other parts of the system model.
- We are working on various types of mode confusion complexity analysis and implications for training.

Executable Specifications as Prototypes

- Easily changed
- At end, have specification to use
- Can be reused (product families)
- Can be more easily reviewed
- If formal, can be analyzed
- Can be used in hardware-in-the-loop or operator-in-the-loop simulations